A Novel Approach For Computer Worm Control Using Decentralized Data Structures

Douglas Roffel (droffel@ucsc.edu), Christopher Garrett (cmgarret@ucsc.edu)

December 13, 2014

Abstract

In this paper, we propose a novel communications module for controlling a computer worm in a way that (if properly executed) completely masks the identity of the operator, allows for rapid distribution of commands, and cannot be realistically censored. This new control technique utilizes the Blockchain, a decentralized data structure, to store and retrieve data related to the operation of the worm.

What is the Blockchain?

The Blockchain is a decentralized data structure, copies of which reside on thousands of computers spread across the Internet. This structure, first described by Satoshi Nakamoto in the Bitcoin whitepaper (1), is a public ledger containing a record of all transactions that have ever been performed on the Bitcoin network. The Blockchain is a linked list of "blocks", each containing thousands of individual transactions, and the hash of the block prior. This ledger is essentially uncensorable: inserting a block into the Blockchain (referred to as "mining") takes a significant amount of computing power, and censoring individual transactions would require an entity to collectively control 51% of the network as a whole. At the current global hashrate of approximately 300,000,000 GH/s, the Bitcoin network has more collective hashing power than the top 500 supercomputers in the world combined, making the task of censorship infeasible without purchasing bulk specialized hardware costing hundreds of millions of dollars.

Technical Implementation of the Blockchain

The Blockchain can be thought of as a linked list. A Bitcoin transaction is the process in which the ownership of a set of Bitcoins is transferred from one address to another. This ownership is implied in the form of a mathematical equation, of the form "Here is a mathematical proof that I own the private key behind this public key. The person who owns these coins is the person who can solve a mathematical equation that proves they control the private key behind another public key."

In this sense, the ownership of coins can be traced all the way back to the block that they were mined in, as a series of changes of ownership (transactions). The Blockchain is the most up-to-date chain of blocks, where "most up to date" is defined as that chain which contains the most blocks, where every block is valid. Transaction hashes are signed by the private key behind the public key, such that in order to verify the transaction from one key to another, one simply needs to decrypt the transaction using the sender's public key. Due to how public key encryption works, this will return the valid transaction hash, proving the owner of the coins decided to transfer them.

The Bitcoin network uses a Proof-of-Work to provide security against multiple spendings of the same Bitcoins. Bitcoin miners, entities who process transactions for the reward of minting new coins every 10 minutes, repeatedly hash the block of transactions, along with an incremented nonce (nonsense counter value), in the hopes of finding a nonce that causes the hash of that nonce and the block of transactions to have a series of leading zeroes proportional to the current difficulty. This gets exponentially harder as the number of leading zeroes increases, and the network self-adjusts so that the difficulty is approximately enough that a block is generated every 10 minutes on average. (1)

Data Storage and Retrieval

A Bitcoin wallet address traditionally has three parts, a version, a hash, and a checksum. For properly encoded addresses, the hash is generated from a private key, ownership of which can be later proven when spending the coins contained within.

While most addresses are created from a known private key, the concept of a "Burn Address" exists, wherein an address with no known private key is generated, in order to provably remove Bitcoins from the ecosystem. The key insight into the design of our communication module was that arbitrary data can be encoded in an outgoing transaction from a target address. Most burn addresses are nonce, however, using this principle, any 20 bytes of data can be stored in the section of the address that usually stores the hash, and a data interpretation "mode" can be set by sending a specific amount of Bitcoin in that transaction.

By sending intelligible data (as opposed to a hash of a public key) in the receiving address of a Bitcoin transaction, the sent coins are effectively burned, and are unable to be sent again (the amount of burned coins is negligible however, as each mode requires only a small amount of Bitcoin to be sent, totalling fractions of a penny at current prices). The "true" cost of sending a command to the worm network is currently less than 3 cents, the per-transaction fee on the Bitcoin Network.

Analysis of Modern Worm Control Techniques

Modern worms are designed in such a way as to give the writer access to the infected computer post-infection. Worms can be classified by the methods through which they are given instructions by their creator, and attempts at damage control are generally centered around cutting off all lines of communication to the worm. In the past, worms have needed a centralized Command and Control server under the control of the creator from which instructions are given, and this is generally considered the easiest place to shut down the network. Newer worms, like Slapper, use Peer-to-Peer networks to communicate, making them much more difficult to shut down, as there is no central location from which commands are issued that can be targeted. While peer-to-peer worms are more difficult to shut down than their centralized counterparts, they are still vulnerable to sybil, eclipse, and poisoning attacks.

Hardcoded Command & Control Server List

The first generation of worms used hardcoded IP addresses of servers under the creator's control. These worms are easily shut down by governments, who contact ISPs in the countries those servers are hosted in, and get the offending servers shut down. Once this happens, the creator no longer has any way of talking to or receiving information from their botnet, effectively shutting it down. Infected computers still exist, and infections may continue unabated, but the worm is subsequently made impotent. A single hardcoded server is a major point of failure, and oftentimes many servers are used, located in different countries, to reduce the ability to shut all of them down at once.

IRC Server

Internet Relay Chat (IRC) servers are lightweight, and can be run nearly anywhere that has a network connection. These features make them a useful tool for worm control, and some modern worms still use them as command and control centers for launching attacks. As infected computers connect to the offending IRC channel, the attackers can additionally harvest and log them, later allowing them to directly connect to the infected computers individually, in the case that the IRC server gets shut down. This method of control requires hardcoding the server IP and channel, and therefore is vulnerable to being shut down by ISP takedowns and blackholing. A modern virus that utilized an IRC command and control server was W32/Tendoolf.

Public Location

Some worms watch a known public location, for example, an internet forum, or a news site comments section, to determine what to do next. These worms generally have a hardcoded public key, and the creator controls the pri-

vate key behind it. When the creator decides to send a command to their botnet, they sign the command with their public key, and post it to the location watched by the worm, which subsequently performs the designated action. Posting to a public location generally carries the risk of censorship by the people who control the public website: the Mac.BackDoor.iWorm virus used this method of control, posting commands to an unused (at the time) Minecraft subreddit, /r/minecraftserverlist. This worm was defanged when the Reddit administrators expunged the offending data from their site, and banned the user posting commands.

Peer-to-Peer Worms

Peer-to-Peer worms are much more difficult to shut down than worms with a centralized command and control server. Rather than receiving their instructions from a centralized location, computers infected with a Peer-to-Peer worms transfer instructions directly to each other. This allows them to propagate instructions in a way that cannot be easily blocked, thus maintaining control of the network.

There are a few forms of defense against these forms of networks. One key strategy is know as a Sybil Attack. This attack works by subverting "proper" peers by faking a majority of the network and confusing the peers. The faked majority is able to confuse the peers by misdirecting them, essentially telling them that computers that are not peers are in fact peers. This confusion reroutes the control structure of the network, causing it to fail. (2, 3)

Advantages Over Current Control Methods

Utilizing a decentralized data structure to store and retrieve information related to a worm's operation provides a system where there is no single point of failure. If implemented properly, the only way to shut down the operation of this module would be to shut down every node in the Bitcoin network that has (and is broadcasting) a full copy of the Blockchain: a task which is completely impractical (not unlike killing a hydra), considering the current difficulty the US government has shutting down even individual websites like The Pirate Bay.

Points of Failure and How to Mitigate Them

This communications module currently relies on a centralized indexer of the blockchain, Blockchain.info. Use of a centralized index is not actually required, in fact, this can be implemented as a thin client that downloads the blockchain, checking each individual block for the existence of a transaction from the current target address as they download, and discarding the blocks afterwards. This minimizes the memory footprint, as each block is currently at most 4mb in size. Other than our current (unrequired) reliance on an indexer, there doesn't appear to be any single location that can be targeted to shut down operation of the module.

Example Payloads

In designing this communications module, we decided to implement a few payloads that might be present in an actual worm, as an example of how versatile a blockchain-controlled worm can be, and how it could be controlled remotely without the need for a central server. We decided to use Rust as the primary language for implementation, with example implementations in shell scripts as well. The latest version of the code is located at:

https://github.com/pzuraq/InChain

DDOS simulation

If a single satoshi (the smallest increment of Bitcoin transferable) is sent, an IP address is decoded from the recipient address, and 100 GET requests are performed. To generate a proper recipient address, the program ./ip2btc can be run, followed by 4 space-delimited octets of an IP address. For example, running

"./ip2btc 192 168 1 1" Would return the address 12kbqw5pyG35tEFXYfNXnRczXoDU7ETns4.

The current target address is this one, which can be verified by running the ./target program, which displays the current target IP address.

Reverse Shell

If two satoshi are sent to an address, the program will open a reverse shell to a target IP, encoded in the recipient address. This allows the controller of the worm to change the command and control server at will, without the risks inherent to a hardcoded server IP. If the current controlling server is compromised, the controller can simply change the server IP to something else.

Change Watch Address

In the event that the controller of a botnet wishes to transfer control of the botnet to someone else (be it to another address they control, or the highest bidder), sending 4 satoshi to any address will turn that address into the new controlling address.

Implications

The design of this communications module has profound implications for decentralized virus control. The ability to store targeting data in a decentralized data structure leads to previously infeasible possibilities for cybercriminals, for example:

Control of botnets can be trustlessly transferred like trading cards between two parties on the Internet for a digital currency such as Bitcoin, and they become much more difficult (if not nigh impossible) to shut down using traditional means. An enterprising cybercriminal could even create a Cryptolocker-esque virus that self destructs after a set ransom is paid into a Bitcoin address by any altruistic individual (or government) who wishes to remove it from the world.

Works Cited

1. Bitcoin: A Peer-to-Peer Electronic Cash System; Satoshi Nakamoto:

   `https://bitcoin.org/bitcoin.pdf`

2. P2P as botnet command and control: a deeper insight; David Dittrich & Sven Dietrich:

   `http://staff.washington.edu/dittrich/misc/malware08-dd-final.pdf`

3. Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm; Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, Felix Freiling:

   `https://www.usenix.org/legacy/event/leet08/tech/full_papers/holz/holz_html/index.html`